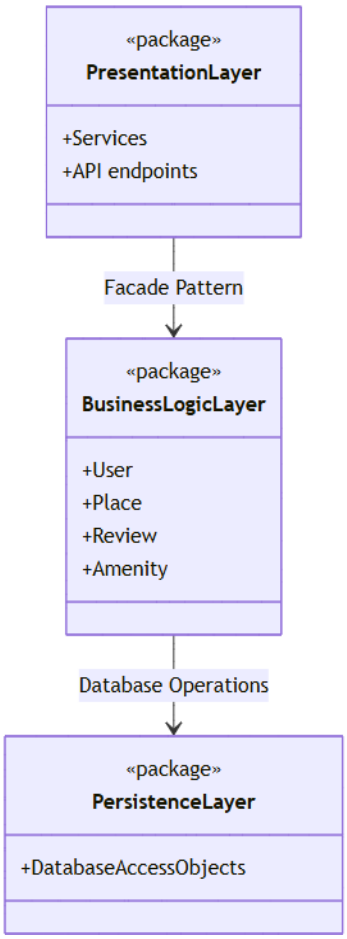
HBnB Architecture Documentation

1. INTRODUCTION

In this project you will see many different diagrams representing the function of our project AirBnb clone named “HBNB”. It represents the architecture of our clone. It contains the:

- High-Level Package Diagram
- Detailed Class Diagram for Business Logic Layer
- Sequence Diagrams for API Calls

2. HIGH-LEVEL-ARCHITECTURE



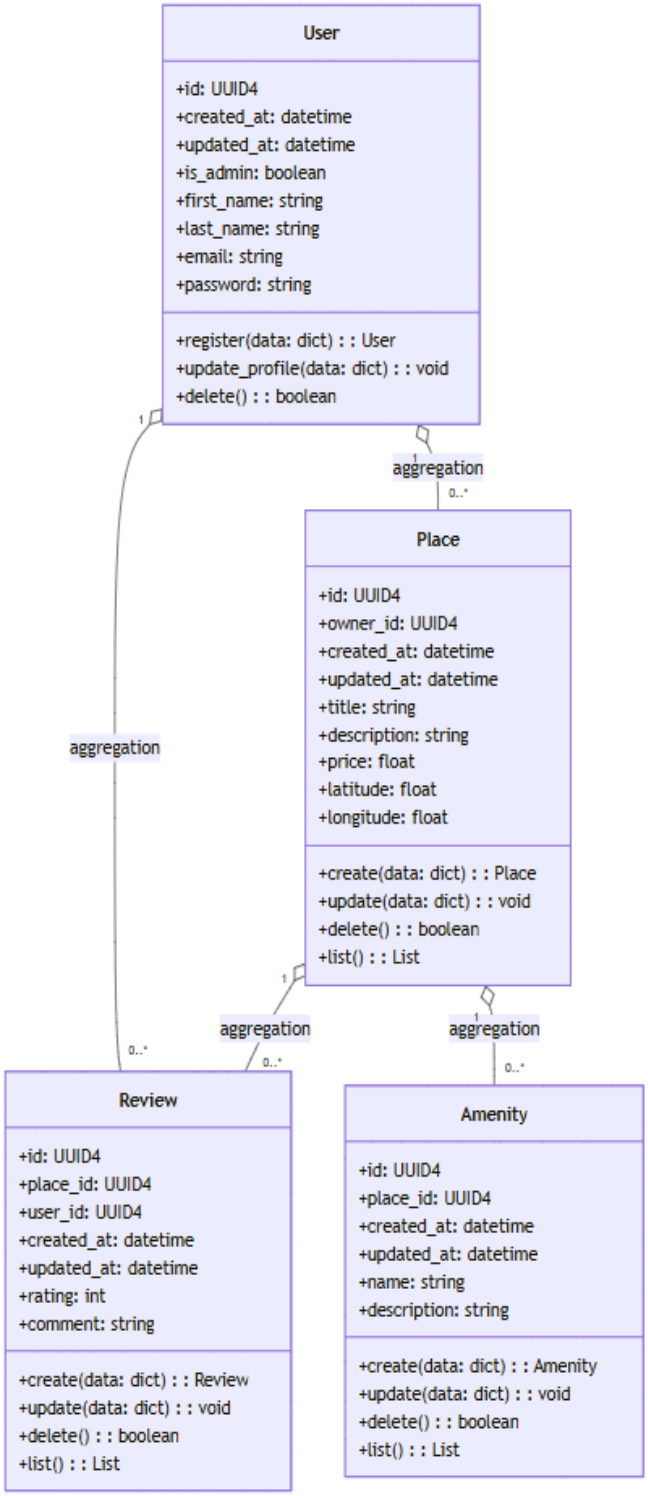
Presentation Layer: Handles all user interactions through services and API endpoints.
It provides a simplified interface for external communication.

Business Logic Layer: Contains the core logic and models (User, Place, Review, Amenity).
It enforces business rules and orchestrates operations between layers

Persistence Layer: Manages data storage and retrieval.
It interacts directly with the database through Data Access Objects (DAOs).

Façade Pattern: The Presentation Layer communicates with the Business Logic Layer via a facade,
providing a unified and simplified interface.
This pattern decouples external services from internal complexity,
promoting modularity and maintainability.

3. BUSINESS LOGIC LAYER



User: Represents a person using the platform.
Includes attributes like name, email, password, and a boolean flag for admin status.
Users can register, update their profile, and be deleted.

Place: Represents a property listed by a user (the owner).
Includes attributes like title, description, price, and location coordinates.
Places can be created, updated, deleted, and listed.

Review: Represents a review posted by a user for a place.
Includes a rating and a comment.
Reviews are linked to both a user and a place.
Supports create, update, delete, and list operations.

Amenity: Represents a feature or facility offered by a place (e.g., Wi-Fi, parking).
Includes a name and description.
Amenities are associated with a specific place and can be managed through standard CRUD operations.

Relationships:

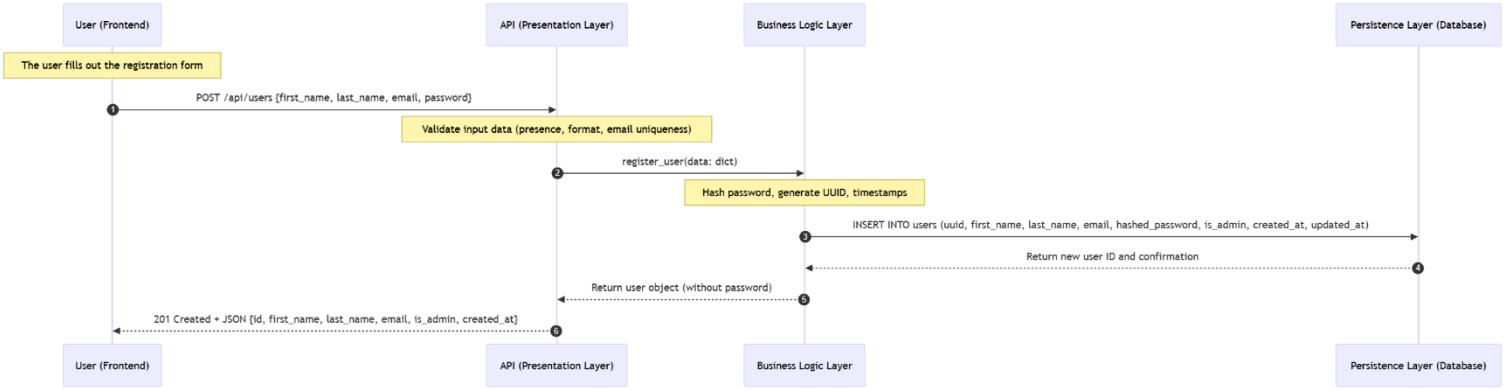
- A **User** can own multiple **Places**
- A **User** can write multiple **Reviews**
- A **Place** can have multiple **Reviews** and **Amenities**.
- All relationships are modeled as aggregations, emphasizing ownership without strong lifecycle dependency.

4. API INTERACTION FLOW

4.1 User Registration

Client -> API -> Logic -> DB -> Logic -> API -> Client

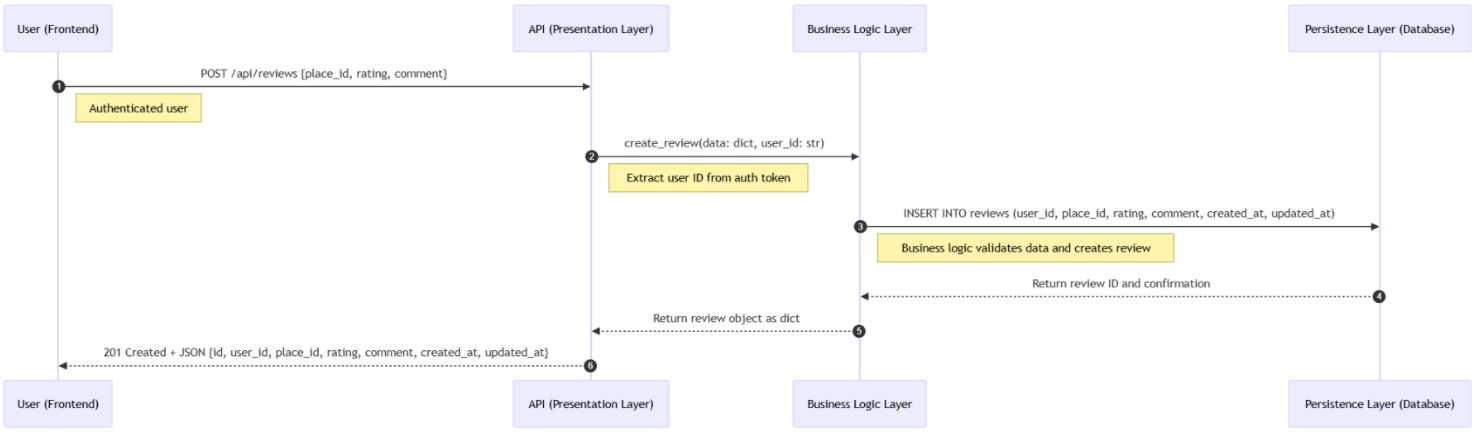
Handles data validation, password hashing, and user creation



4.2 Place Creation

Client -> API -> Logic -> DB -> Logic -> API -> Client

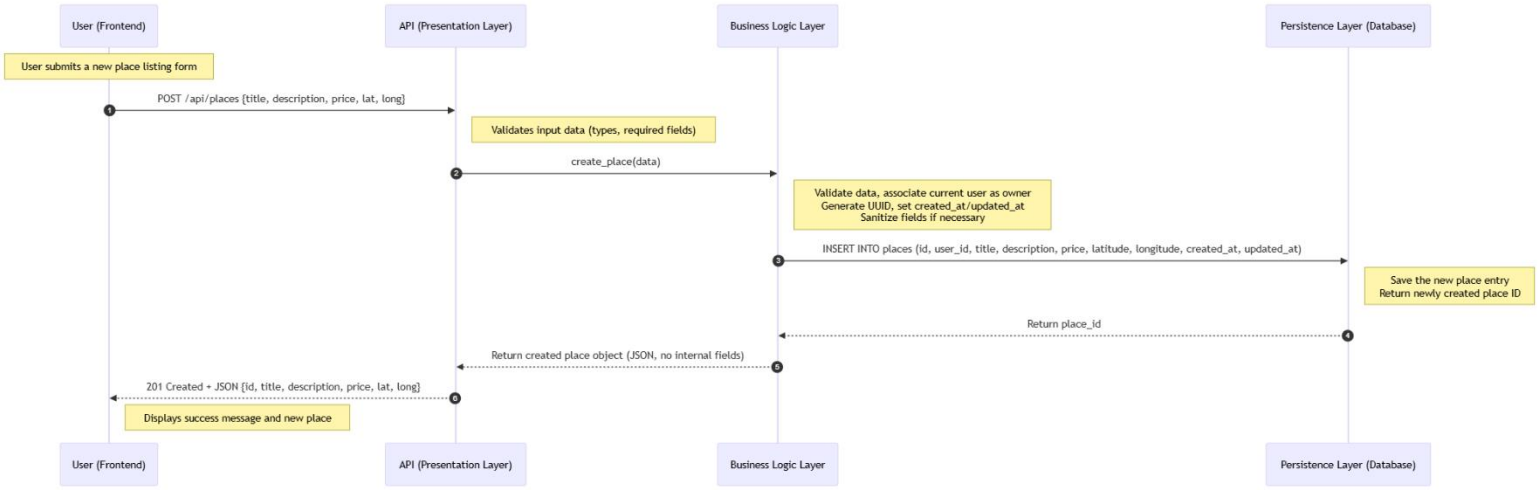
Creates a place with user association and stores it in DB



4.3 Review Submission

Client -> API -> Logic -> DB -> Logic -> API -> Client

Validates and creates a new review



4.4 Fetch Places

Client -> API -> Logic -> DB -> Logic -> API -> Client

Fetches filtered places and associated reviews and amenities